Videojuegos

Curso de Diseño y Programación

Nº 7 5,99 euros



Visualizador
de terrenos y de modelos

Gráficos 2D
y sprites

Editar audio
con ***Sound Forge***

7







Curso práctico de Videojuegos

AUTOR DE LA OBRA
Marcos Medina
DIRECCIÓN EDITORIAL
Eduardo Toribio
etoribio@iberprensa.com
COORDINACIÓN EDITORIAL
Eva-Margarita García
eva@iberprensa.com
DISEÑO Y MAQUETACIÓN
Antonio Gª Tomé
PRODUCCIÓN
Marisa Cogorro
SUSCRIPCIONES
Tel: 91 628 02 03
Fax: 91 628 09 35
suscripciones@iberprensa.com
FILMACIÓN: Fotpreim Duvial
IMPRESIÓN: Gráficas Don Bosco
DUPLICACIÓN CD.ROM: M.P.O.
DISTRIBUCIÓN
S.G.E.L.
Avda. Valdelaparra 29 (Pol. Ind.)
28108 Alcobendas (Madrid)
Tel.: 91 657 69 00

EDITA: Iberprensa
www.iberprensa.com

CONSEJERO
Carlos Peropadre

REDACCIÓN, PUBLICIDAD Y ADMINISTRACIÓN
C/ del Río Ter, 7 (Pol. Ind. "El Nogal")
28110 Algete (Madrid)
Tel.: 91 628 02 03
Fax: 91 628 09 35
(Añada 34 si llama desde fuera de España.)

DEPÓSITO LEGAL: M-35934-2002
ISBN: Coleccionable: 84 932417 2 5
Tomo 1: 84 932417 3 3
Obra Completa: 84 932417 5 X

Aprende divirtiéndote

Bienvenidos a **Programación y Diseño de Videojuegos**, la primera obra coleccionable cuyo objetivo es formar al alumno en las principales técnicas relacionadas en el desarrollo completo de un videojuego.

A lo largo de la obra el lector aprenderá programación a nivel general y a nivel específico con ciertas herramientas y lenguajes, aprenderá a trabajar con aplicaciones de retoque de imagen y también de diseño 3D y animación. Descubrirá las aplicaciones profesionales más importantes de audio y conocerá la historia de lo que se denomina "la industria del videojuego", los últimos 20 años, los juegos que marcaron un avance, sus creadores y en general la evolución del videojuego.

Pero además, esta obra tiene un segundo objetivo, desarrollar y potenciar la creatividad del lector, nosotros a lo largo de las diferentes entregas pondremos las bases y tú pondrás tu ingenio, tu creatividad y tu capacidad de mejorar.

Comienza aquí un viaje de 20 semanas articulado en 400 páginas y 20 CD-ROMs cuya finalidad es proporcionar las bases mínimas para después cada uno continuar su camino.

Recuerda que para alcanzar el éxito necesistas cumplir tres condiciones: que te gusten los juegos, poseer cierta dosis de creatividad y finalmente capacidad de estudio.

Una la cumples seguro.

sumario

7

PARA ENCUADERNAR LA OBRA:

- *Para encuadernar los dos volúmenes que componen la obra "Programación y Diseño de Videojuegos" se pondrán a la venta las tapas 1 y 2.*
- ***Tapas del volumen 1 ya a la venta.***
- ***Los suscriptores recibirán las tapas en su domicilio sin cargo alguno como obsequio de Iberprensa.***

SERVICIO TÉCNICO:
Para consultas, dudas técnicas y reclamaciones Iberprensa ofrece la siguiente dirección de correo electrónico: games@iberprensa.com

PETICIÓN DE NÚMEROS ATRASADOS:
El envío de números sueltos o atrasados se realizará contra reembolso del precio de venta al público más el coste de los gastos de envío. Pueden ser solicitados en el teléfono de atención al cliente 91 628 02 03

Construcción de herramientas propias para el desarrollo del **juego**

En este número haremos un inciso antes de comenzar a programar el juego. Generalmente es muy común el desarrollo de otro tipo de aplicaciones que ayuden a los grafistas y programadores en el proceso de construcción de un juego. Nos referimos a pequeños programas que sirvan para chequear gráficos.

En nuestro caso, vamos a desarrollar un visualizador de terrenos y otro de modelos. También realizaremos una pieza importante en el conjunto de juegos cuyos mapeados se basan en zonas diferentes: los editores de niveles. Ya que en "Zone of Fighters" la acción transcurre en zonas de combate, desarrollaremos un editor para crearlas a gusto del jugador. Esta herramienta ayudará a alargar la vida del juego y añadirá un atractivo más para el consumidor.

(■) VISUALIZADOR DE TERRENOS

Con esta aplicación, podremos generar un terreno a partir de un mapa de alturas, aplicarle texturas y recorrerlo con la cámara. Básicamente, así será nuestro visualizador de terrenos. Sin embargo, habrá elementos en este número que no hemos visto aún, aunque esto no significa ningún problema; lo importante es que se entienda la estructura básica de esta aplicación. De todas formas, todo el código contenido en el fichero "vis_terreno.bb" del CD está comentado y lo que no se entienda ahora se comprenderá perfectamente más adelante.

El núcleo de este visualizador, y del de modelos, nos servirá para realizar el editor de zonas de combate. En el CD-ROM puedes ver una imagen con la estructura del programa en un sencillo pseudocódigo.

Observamos en esta estructura que el programa está dividido en dos partes. Una primera que constituye el núcleo principal encerrado en un bucle, y una segunda parte compuesta por las distintas funciones o tareas que se realizan. Vamos a explicar los procesos más significativos del código. La mayoría de las instrucciones las hemos visto en la sección de Blitz3D. Es conveniente que tengas el código a mano mientras lees las explicaciones.

Antes de entrar en el bucle principal que gobierna el programa, debemos definir el modo gráfico y activar el doble búfer:

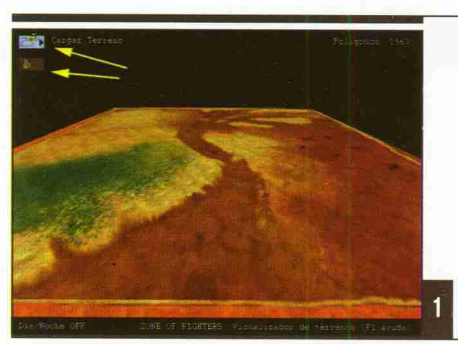
```
Graphics3D 640,480,16,1
SetBuffer BackBuffer()
```

Prepararemos el entorno donde se colocará el terreno y definiremos todas las constantes, variables y tipos utilizados.

La interfaz de usuario constará de dos iconos en la parte superior de la pantalla que servirán para cargar un terreno o salir del programa (Fig. 1).

Para ello precisaremos de un tipo que defina el gráfico, coordenadas y un literal con el nombre de la acción que representa. Aunque sólo tenemos dos, esta estructura la usaremos luego para el editor de zonas donde tendremos muchos más.

```
Type botones
    Field grafico
    Field x,y
    Field nombre$
```



1

La interfaz de usuario del visualizador de terrenos está compuesta por dos iconos en la parte superior de la pantalla.

```
End Type
Dim boton.botones(1)
Dim botonB.botones(1)
```

La matriz nos servirá para añadir más botones si lo necesitáramos más adelante. Definimos *botonB* porque será el icono de otro color que se mostrará cuando estemos encima con el cursor. A continuación, cargamos la imagen del cursor y definimos el resto de variables globales utilizadas en el programa. Antes de entrar en el bucle principal, colocamos el cursor del ratón en el centro de la pantalla *(MoveMouse 320,240)* y llamamos a la función para crear las luces y la cámara (luego explicaremos cada una de las funciones detenidamente). En la línea:

```
Collisions ENTIDAD_CAMARA,
ENTIDAD_TERRENO,2,3
```

Llamamos a la función de colisión del Blitz3D para que la cámara detecte al terreno y se deslice por él. Empezamos el bucle principal borrando el búfer actual de dibujo (la pantalla) con *Cls*, seguidamente detectamos desde el teclado las opciones: salir, activar o desactivar la niebla, activar el

paso del día a la noche y convertir el terreno en modo wireframe (en alambre, sin texturas). El sistema de *Switch* nos ayudará a conmutar valores para activar o desactivar acciones, simplemente se trata de definir variables cuyo valor será 0 ó 1.

Por ejemplo, en la línea:

```
If KeyDown(4)
SW_DiaNoche=1-SW_DiaNoche
```

Si pulsamos la tecla "3" el valor de SW_DiaNoche valdrá 1 menos su antiguo valor:

Si SW_DiaNoche=0

```
SW_DiaNoche=1 - 0 ‡ 1
```

Si por el contrario vale 1

```
SW_DiaNoche=1 - 1 ‡ 0
```

De esta forma, con una sola instrucción podemos asignar un 0 o un 1 al Switch SW_DiaNoche

Para controlar la rotación de la cámara utilizaremos el ratón; así que, mientras pulsemos el botón derecho podremos moverla.

Para girar la cámara mediante el ratón utilizamos una serie de variables que almacenarán la velocidad de movimiento horizontal (eje X) o vertical (eje Y) de éste.

En mx y my almacenamos la velocidad relativa y la convertimos a otra absoluta:

```
mx#=mx#-Float(MouseYSpeed())/
-7 : mx=mx/1.5
my#=my#+Float(MouseXSpeed())/
-7 : my=my/1.5
```

Esto nos servirá para sumar grados al giro en las variables X e Y

```
X#=EntityPitch(camara)+mx
```

Giro en vertical (EntityPitch) + los grados del movimiento del ratón (mx)

```
Y#=EntityYaw(camara)+my
```

Giro en horizontal (EntityYaw) + los grados del movimiento del ratón (my)

A continuación, limitamos el giro vertical a sólo 180 grados en los dos sentidos

```
If X>=89 And X<=180 X=89
If X<=-89 And X>=-180 X=-89
```

Seguidamente, hacemos efectivos los giros de ambos ejes y volvemos a colocar el cursor del ratón en la mitad de la pantalla para no perder el desplazamiento continuo de éste.

Después de actualizar y dibujar el mundo 3D y antes de intercambiar los búferes, debemos dibujar y detectar la colisión del cursor con los iconos, así como la impresión de los textos de información y ayuda en pantalla.

```
For n=0 To 1
  DrawImage boton(n)\grafico,
  boton(n)\x,boton(n)\y
  If ImagesOverlap(cursor,
  MouseX(),MouseY(),boton(n)\
  grafico,boton(n)\x,boton(n)\y)
    DrawImage botonB(n)\grafico,
    boton(n)\x,boton(n)\y
    Text 60,boton(n)\y,
    boton(n)\nombre$
    If MouseDown(1)
    eleccion=n Else eleccion=-1
  EndIf
```

Con este bucle dibujamos los iconos y posteriormente utilizamos una sentencia "If" para detectar la colisión de éstos con el cursor. Si hay colisión, cambiamos de icono y si además pulsamos el botón izquierdo, guardamos en "elección" el número del icono detectado.

Vamos a explicar las distintas funciones que nos encontramos en el programa. En primer lugar, en la función *Crea_Entorno()* asignamos una intensidad determinada a la luz ambiental, ésta afectará a todas las luces que creemos a partir de este momento. Después, creamos el Sol, como luz direccional, con un color determinado y con una inclinación de 45 grados.

Posteriormente, debemos crear una cámara para poder ver el entorno 3D y la sumamos al mapa de colisiones. Asignamos un color de fondo y el mismo color para la niebla, y posicionamos la cámara para que apunte hacia el terreno y lo encuadre en la pantalla.

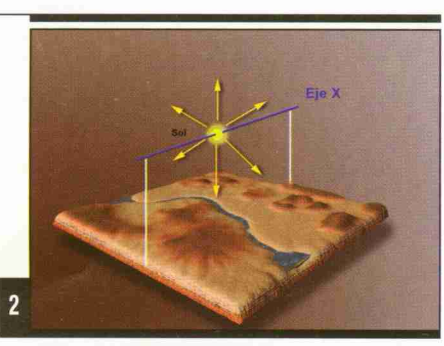
En la siguiente función *Cargar_Terreno()*, obtenemos el nombre del mapa de alturas y de las texturas del terreno para poder crearlo y asignarle el aspecto. Disponemos de tres bucles *"Repeat ... Until"*, uno para cada petición. En cada uno de ellos colocamos una instrucción *Input* que obtendrá el nombre del fichero de manos del usuario. Además, con la función *FileExists()*, nos aseguramos que dicho fichero existe antes de seguir. Continuamos, borrando el terreno que esté en pantalla; si no hiciéramos esto, al crear otro, se sumaría al anterior. Utilizamos un Switch (SW_carga) para saber si el terreno que vamos a crear es o no el primero, porque si borramos sin tener nada creado el programa dará un error. Para terminar, creamos el terreno a partir del mapa de alturas, lo sumamos a la lista de colisiones y le asignamos 2000 polígonos para su creación. Además, activamos el sistema LOD (nivel de detalle) y el sombreado.

Para finalizar, le asignamos una primera textura que lo cubrirá completamente y otra mezclada de forma tileada para proporcionar más detalle.

En la función *Salir()*, nos aseguramos de que el usuario desea salir, en caso contrario regresamos al bucle principal y si es correcto finalizamos el programa *(End)*.

La función *Dia_Noche()* es la que se encarga de disminuir y aumentar la intensidad de luz ambiental en:

```
If luz#<10 variacion_luz=
(variacion_luz)*-1
If luz#>254 variacion_luz=
```



Un esquema gráfico de cómo se mueve la luz del sol en el visualizador de terrenos para generar el paso del día a la noche.

```
(variacion_luz)*-1
luz#=luz-variacion_luz#
```

Las sentencias anteriores se basan en que cualquier valor positivo multiplicado por –1 se convierte en negativo y si el valor es negativo se convertirá en positivo. De este modo, realizamos estas operaciones con la variable *variacion_luz* para el cambio de intensidad de luz.

Continuamos, rotando la luz en su eje X para simular el movimiento del Sol (Ver Fig. 2).

(■) VISUALIZADOR DE MODELOS

La estructura de esta aplicación es exactamente igual que la anterior. La única diferencia es que en vez de cargar mapas de alturas y crear terrenos, cargamos objetos 3D (Fig. 8).

Ahora esta operación la realizamos en la función *Carga_Modelo()*. Aquí, una vez cargado el modelo y su textura, detectamos de qué tipo es según la extensión del fichero.

```
If Right(modelo$,3)="3ds"
  entidad_modelo=LoadMesh(modelo$)
EndIf
...
```

En caso de ser un archivo .3DS, simplemente lo cargamos, le asignamos la textura y lo posicionamos delante de la cámara. Por el contrario, si se trata de archivos con animación (bien .B3D o .MD2) cargamos el modelo, lo animamos, le asignamos la textura y lo posicionamos delante de la cámara.

(■) EDITOR DE ZONAS

Esta herramienta es muy útil para diseñar niveles con el mínimo esfuerzo. Básicamente, la idea es colocar, a nuestro gusto, los objetos de decorado, animales y plantas sobre un terreno determinado y grabar los datos en un fichero. Este archivo será posteriormente leído por el juego, el cual volverá a colocar todos los elementos sobre el terreno en la misma posición.

El éxito o no de nuestro editor dependerá de su facilidad de manejo. Para ello, vamos a proporcionar una vista totalmente tridimensional del terreno al usuario, el cual podrá moverse libremente por él y colocar o editar los diferentes elementos con el ratón *in situ*. De esta forma, tendrá una visión exacta de cómo se verá su nivel en el juego.

La estructura básica del programa se inspira en los dos visualizadores anteriores. En el CD-ROM podemos ver un pseudocódigo de su funcionamiento.

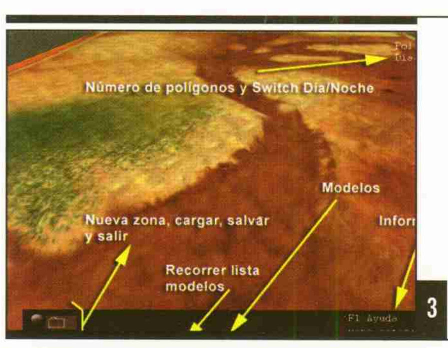
Vamos a explicar sólo las partes más importantes del código. De todas formas, en el CD-ROM se halla también el código fuente totalmente comentado ("editor.bb")

Lo fundamental del programa es que su funcionamiento se basa en el manejo de dos estructuras principales. Una primera:

```
Type tipo_mesh
     Field x#,y#,z#,ry#
     Field entidad
     Field Objeto
     Field tipo_entidad$
End Type
```

Es utilizada para manejar todos lo objetos que se están colocando y editando para crear el nivel y una segunda estructura con campos ligeramente diferentes, para crear el archivo definitivo que contendrá el nivel o zona de combate:

```
Type Tipo_archivo
     Field Tipo_entidad$
     Field entidad
```



En la imagen se muestra una descripción de los elementos del interfaz de usuario del editor de zonas.

```
     Field Identi_entidad#
     Field x#,y#,z#
     Field ry#
End Type
```

Para crear muchos objetos con la misma estructura disponemos de la matriz:

```
Dim mesh.tipo_mesh
(numero_maximo_mesh)
```

Esta matriz de objetos la controlamos con dos índices: *indice* e *indice_mesh*. El primero lo utilizamos en el bucle principal para pasarle a la función *Coloca_objeto()*, información acerca de qué objeto hemos seleccionado *(Item3D(indice))* y de cómo se llama.

Hemos creado una interfaz gráfica más completa con información en pantalla acerca de los objetos y la posibilidad de elegirlos con el ratón de forma visual, mediante dos iconos de flechas (Ver Fig. 3).

Una vez que hemos elegido con el ratón sobre el objeto que queremos colocar, pasamos a la función que nos permitirá realizar esta operación: *Coloca_objeto()*. A esta función llegamos con la información que nos ha proporcionado la variable *indice* sobre el objeto que vamos a colocar. El primer valor m nos servirá para asignarle el mesh (la forma) correspondiente al nuevo objeto:

```
mesh(indice_mesh)=
New tipo_mesh
mesh(indice_mesh)\entidad=
CopyEntity(m) fl
```

y el valor obj para poder identificar el nuevo objeto dándole un nombre:

```
Select obj fl
  Case 0: mesh(indice_mesh)\
tipo_entidad$="almacen1"
  Case 1: mesh(indice_mesh)\
tipo_entidad$="almacen2"
  ...
```

A continuación aplicamos la función mágica que nos permitirá mover el objeto sobre el terreno 3D mediante el ratón: CameraPick:

```
CameraPick(camara,MouseX(),
MouseY())
mesh(indice_mesh)\x#=PickedX()
mesh(indice_mesh)\z#=PickedZ()
```

PickedX() y *PickedZ()* nos proporcionan también las coordenadas exactas del terreno donde tenemos el cursor.

Como ya veremos, esta función sólo es válida si hemos asignado con anterioridad el terreno en modo "pickeable" (desmontado en polígonos) con la función *EntityPickMode*.

Después de los comandos típicos para mover el objeto con precisión mediante las teclas, detectamos si hemos hecho clic con el ratón para saber si el objeto lo colocamos definitivamente o no. En caso afirmativo, añadimos el nuevo objeto con los datos que nos interesan para su lectura posterior en el juego e incrementamos el índice global de la matriz de objetos para situarnos en uno nuevo.

```
If MouseHit(1)
  Archivo.Tipo_archivo=
  New Tipo_archivo
  Archivo\Tipo_entidad$=
  mesh(indice_mesh)\
  tipo_entidad$
  Archivo\Identi_entidad#=
  mesh(indice_mesh)\entidad
  Archivo\entidad=
  mesh(indice_mesh)\Objeto
  Archivo\x#=EntityX
   (mesh(indice_mesh)\entidad)
  Archivo\y#=EntityY
   (mesh(indice_mesh)\entidad)
  Archivo\z#=EntityZ
   (mesh(indice_mesh)\entidad)
  Archivo\ry#=mesh
   (indice_mesh)\ry#
  pick=0
  coloca=0
  indice_mesh=indice_mesh+1
EndIf
```

La función *Modificar_objeto()* parece algo complicada, pero en realidad es bien sencilla. Lo que hacemos es recorrer mediante un bucle *For ... Next* la matriz principal de objetos para buscar con cuál de ellos ha contactado el cursor en el terreno. Para aislarlo, recorremos con *For ... =Each* la estructura de cada uno de los objetos de esta matriz. Para ganar velocidad en el proceso de edición, reducimos la búsqueda sólo en los objetos que se encuentran en la vista de la cámara, con la siguiente condición:

```
If EntityInView(mesh(n)\
entidad,camara)=True
```

Además, debemos convertir el objeto en "pickeable" para poder identificarlo cuando lo detecte el cursor. Una vez identificado, se le desactiva el modo "pickeable". De esta forma se gana más velocidad en el proceso:

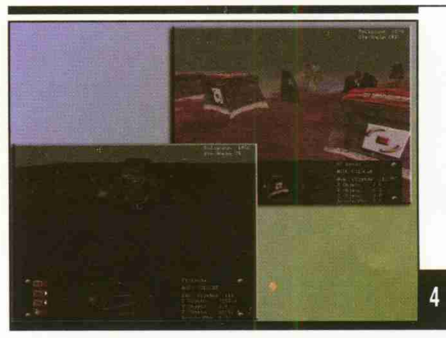
```
EntityPickMode mesh(n)\
entidad,3 fl Acivar
Entidad=CameraPick
(camara,MouseX(),MouseY())
EntityPickMode mesh(n)\
entidad,0 fl Desactivar
```

Para finalizar, haya cambios o no en el objeto, lo actualizamos en la estructura del archivo.

Para crear una zona nueva debemos borrar todos los objetos creados de la matriz de objetos y de la estructura del archivo de zona (función *Nueva_Zona*). Si lo que queremos es grabar la zona, simplemente damos un nombre para crear el fichero en el disco y grabar campo por campo todos los objetos contenidos en la estructura *Tipo_Archivo* (función *Grabar_Zona*). Después, inicializamos la cámara para obtener una vista general del terreno y así poder sacar una instantánea de la zona (captura de pantalla), que utilizaremos en el juego para la elección del nivel por el usuario.

Y concluimos, con la función *Cargar_Zona()*, que simplemente lee todas las estructuras de objetos contenidas en el fichero, luego se va colocando en la matriz de objetos y, finalmente, se posiciona en el terreno.

Hemos tenido un duro trabajo en esta entrega, pero a medida que avancemos en los conocimientos sobre Blitz3D, será muy fácil adaptar estas aplicaciones a vuestras necesidades. Una buena terapia de perfeccionamiento y comprensión es hacer muchas pruebas modificando el código y así poder estudiar diferentes comportamientos en las funciones. Hay que prestar una especial atención al manejo de estructuras de datos, de las cuales se ha podido observar su importancia en las herramientas que hemos visto.



4

Dos bonitos ejemplos que muestran la posibilidad de recorrer con la cámara todos los rincones de la zona editada.

En el próximo número...

... vamos a entrar en el desarrollo del código de "Zone of Fighters". Empezaremos creando una estructura global del juego mediante pseudocódigos, importantísima para llevar un buen control en esta importante empresa.

Grafismo 3d. Modelado de la bionave de combate

Por fin ha llegado el momento de convertir en un modelo 3D el diseño de la bionave de combate. Para ello, vamos a modelar con el programa MilkShape 3D por el método de primitivas base, procurando mantener un número de polígonos razonable.

Antes de empezar vamos a organizar nuestros ficheros. Por ejemplo, creemos una carpeta principal en el disco duro llamada "juego_ZOF". Ahí crearemos las demás carpetas temporales que guardarán todo el trabajo. Al terminar, los ficheros finales resultantes se pasaran a una carpeta definitiva que usaremos en la instalación del juego. Dentro de esta carpeta creamos otra llamada "Bionave" y dentro de esta, otra llamada "Modelado".

Al final resultaría: "C:\juego_ZOF\Bionave\Modelado".

Dividiremos nuestro trabajo en partes. En la figura 1 podemos ver un esquema de las distintas piezas que forman la bionave.

CUERPO DE LA NAVE

La estructura general de la bionave es rectangular, luego se le añaden los demás elementos como las alas, alerones, cañones, etc. El cuerpo principal está dividido en dos partes, una sobre la otra. Para la parte inferior partimos de una caja a la que le damos la forma rectangular con la opción de escalado (F4). A continuación vamos a seleccionar y mover los vértices del plano inferior hacia dentro para estilizar las formas como se muestra en la figura 2.

Realizamos la misma operación pero a la inversa para la parte superior.

Una vez que tengamos las dos partes juntas, debemos unirlas físicamente por los vértices con la opción "Snap Together" (CTRL+N).

Seleccionamos los vértices de la esquina de ambas partes con la opción "Ignore Backfaces" seleccionada y unimos (Ver Fig. 3).

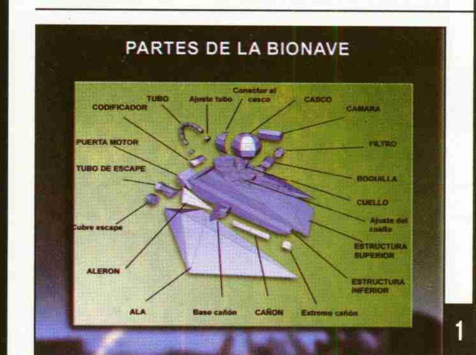
AÑADIR LAS ALAS

Una vez modelado el cuerpo principal de la bionave, iremos creando y añadiendo el resto de las piezas. Seguiremos con las alas. Éstas tienen forma de flecha y se pueden formar perfectamente a partir de un rectángulo. Una vez creado éste, seleccionamos los dos vértices exteriores traseros como se muestra en la figura 4 y lo unimos con CTRL+N.

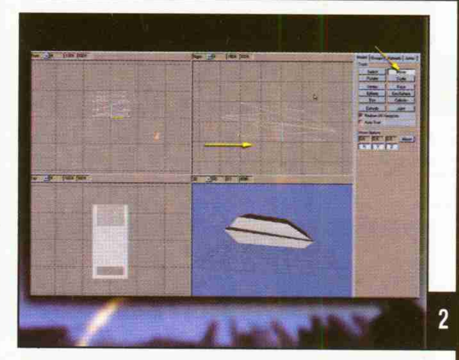
A continuación, con la opción "move" (F2), los desplazamos hacia el exterior, hasta obtener la forma deseada (Fig 4). Haremos las mismas operaciones con los demás vértices. Cuando hayamos terminado con una de las alas, la duplicamos con CTRL +D y hacemos un "Mirror Left <-> Right" (menú "Vertex") para crear y colocar la otra ala.

ESCAPE Y CUBRE ESCAPE

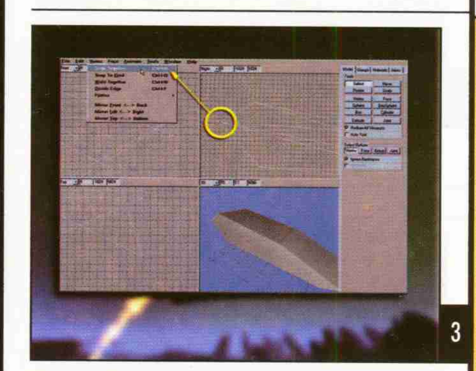
Para el tubo de escape, partimos de un cilindro de dos "Stacks" (partes) y ocho "Slices" (divisiones). Seleccionamos todos los vértices ("Ignore Backfaces" deseleccionado) de la parte central y los desplazamos



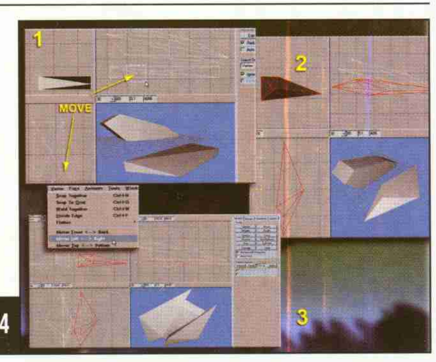
Una gráfica de las distintas partes que componen la bionave de combate.



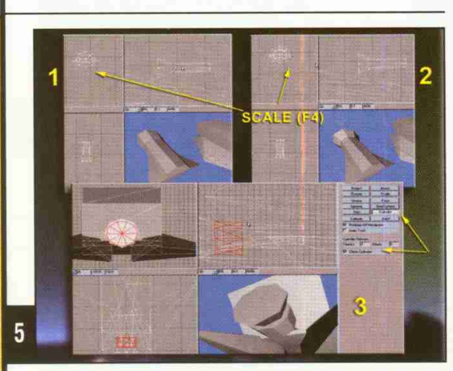
Primer proceso para crear el cuerpo de la nave.



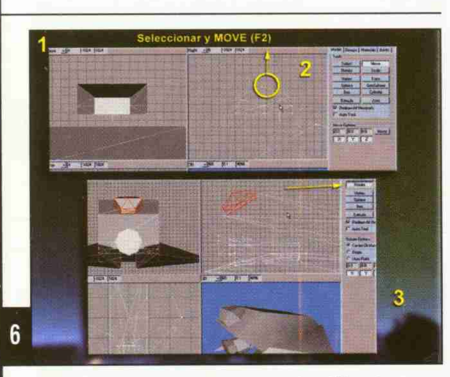
Uniendo las dos partes del cuerpo de la bionave.



A partir de un rectángulo se puede obtener las alas con sólo desplazar los vértices.



Para crear objetos como el escape a partir de cilindros, es importante escalar y mover vértices.



Un sencillo proceso para crear el codificador.



El proceso para obtener la base del cañón puede resultar algo complicado, pero lo importante es obtener una forma en "L".

hacia el lado del final del escape. La forma de éste la realizamos mediante el escalado de vértices seleccionados como se muestra en la figura 5.

Con este procedimiento de escalado obtendremos el aspecto final. Para terminar, rotamos (F3) el escape 90 grados en la vista "Right" y lo situamos en el extremo inferior trasero de la bionave. Para el cubre-escape creamos un cilindro de una parte y diez divisiones, rotamos de igual forma y lo colocamos (ver Fig. 5).

CODIFICADOR

El codificador es la pieza que traduce toda la información que viaja a través del tubo desde la cabeza del luchador hacia la bionave. Para realizarla, partiremos de un cilindro estrecho de seis divisiones en el que desplazamos hacia el exterior uno de los vértices centrales. Realizado esto, rotamos la pieza y la encajamos en la parte trasera de la bionave (ver Fig. 6).

CAÑONES

La bionave posee dos cañones situados a ambos lados. Están formados por tres piezas: un soporte, el cilindro del cañón y el extremo. El soporte tiene forma de "L" y parte de un cilindro de seis divisiones y un "stack", al que se le han desplazado los vértices como se muestra en la figura 7.

Una vez adquirida la forma se duplica (CTRL+D) y se le realiza la función "Mirror Left <-> Right". Para el cuerpo del cañón, basta con un cilindro de dos partes y tres divisiones, al cual alargamos hasta obtener la longitud adecuada. El extremo se realiza exactamente de la misma manera, aunque a partir de un cilindro con una sola parte para ahorrar polígonos. Terminamos realizando las misma operaciones de copiado y espejo para crear el otro cañón (Fig. 8).

AJUSTE DEL CASCO Y DEL CUELLO

Estas piezas situadas en la parte superior del vehículo son la base de la cabeza del protagonista. Ambas están formadas por cilindros de diez divisiones y una parte debidamente rotadas y colocadas (ver Fig. 9).

LA TAPA DEL MOTOR

Ésta es una pieza relativamente pequeña que está situada en la parte trasera sobre el tubo de escape. Partimos de una caja a la que le desplazamos los vértices como se muestra en la figura 10.

CASCO Y CONECTOR

Éstas son las piezas que engloban a la cabeza del conductor de la bionave. Están formadas a partir de sendas esferas de seis partes por seis divisiones. El conector cubre la mitad trasera del casco. Para realizarlo, debemos dividir una esfera en dos, seleccionar la mitad de sus vértices con "Ignore Backfaces" deseleccionado y borrar. Seguidamente, desplazamos la mitad restante hasta cubrir la parte trasera del casco (Fig. 11).

FILTRO DE AIRE Y CÁMARA ESPECIAL

En el juego, la cámara permite aumentar el rango de visión del casco y básicamente está modelado a partir de un cilindro achatado de cinco divisiones y una parte.

El filtro está situado en el casco y sirve para filtrar el polvo y demás elementos nocivos del viciado aire de los terrarios. Está formado por la boquilla y un filtro a cada lado. Para la boquilla creamos un cilindro de ocho partes, seleccionamos los tres vértices exteriores (cuatro caras) y los desplazamos (F2) en la

vista "Right" un poco hacia fuera. Añadimos los filtros, modelando uno de ellos a partir de un cilindro igual a la boquilla pero con uno de sus lados más anchos (seleccionar vértices y escalar en la vista frontal). Para terminar, duplicamos y realizamos un "Mirror" (Fig. 12).

TUBO Y PIEZAS DE AJUSTE

Para las piezas de ajuste del tubo al casco y al codificador, dispondremos de pequeños cilindros con ocho divisiones. Después de escalarlos y rotarlos adecuadamente, los colocamos en los lugares que se muestran en la figura 13.

El tubo de comunicación entre la nave y el piloto tiene forma curva y está formado por ocho cilindros con ocho divisiones y una parte cada uno. Es suficiente con crear uno de estos cilindros. Posteriormente, basta con duplicarlo (CTRL+D) y desplazarlo hacia arriba. La forma curva la iremos construyendo por medio de la rotación de los cilindros que vayamos duplicando. Para rotar activamos en "Rotate Options" la opción "User Point" (punto de usuario) para poder girar el cilindro a partir del puntero del ratón (Fig. 14).

Otra forma de crear el tubo sería uniendo cada cilindro entre sí por medio de los vértices de sus extremos. Es una tarea más laboriosa pero el resultado es más perfecto. El problema radica en que hay que tener todos los cilindros correctamente orientados antes de unirlos, y así conseguir la forma curva adecuada. Creamos un cilindro de ocho divisiones, lo duplicamos, rotamos y desplazamos como en el proceso anterior. A continuación, con "ignore Backfaces" deseleccionado, seleccionamos los vértices de los extremos de ambos cilindros y los unimos con "Snap Together" (CTRL+N).

Volvemos a seleccionar "Ignore Backfaces" para elegir los vértices que quedan por detrás de la vista, y unimos. Realizamos las mismas operaciones para todos los demás vértices. Para completar el ensamblaje, necesitaremos cambiar a la vista frontal para unir los vértices ocultos más difíciles.

Y por último, cambiamos a "wireframe" la vista 3D para descubrir que falta por unir los vértices centrales del cilindro.

Aunque cada cilindro esté unido, siguen siendo piezas independientes en el conjunto del tubo.

AGRUPANDO PIEZAS

Milkshape3D permite la opción de nombrar cada pieza del modelo y formar grupos con ellas. Esta posibilidad resulta muy útil cuando se está trabajando con muchas partes. Antes de realizar las operaciones de agrupamiento hay que renombrar cada pieza. Para ello debemos irnos a la pestaña "Groups" de las opciones de edición. Observaremos una ventana que muestra una lista de todas las partes de la bionave.



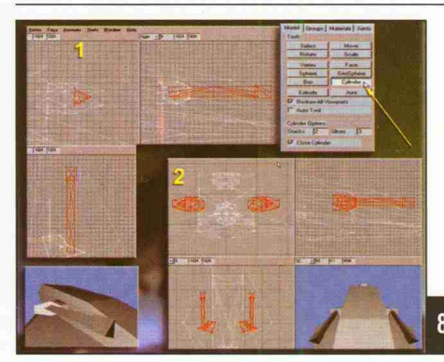
TRUCO

Para evitar que los cilindros no encajen bien, una buena opción sería rotar una vez realizada la duplicación y posteriormente desplazar hacia arriba como se muestra en la figura 15.

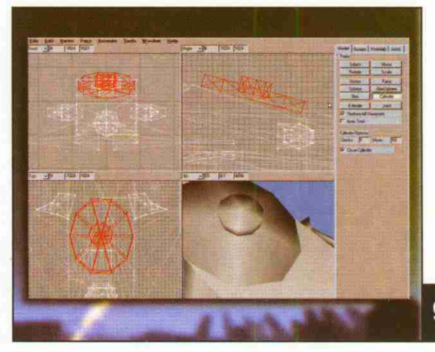


TRUCO

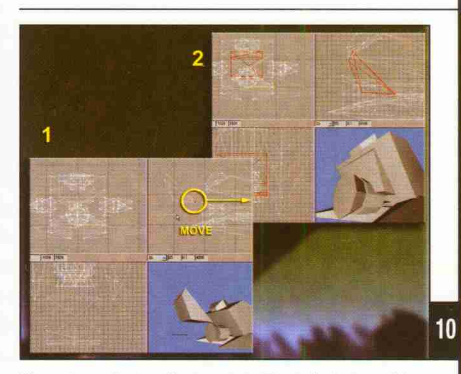
Si al terminar de unir todos los cilindros no se consigue la forma de la curva adecuada, una buena solución sería seleccionarlos individualmente y rotarlos hasta conseguir la curvatura. Al estar soldados, se desplazará también la unión.



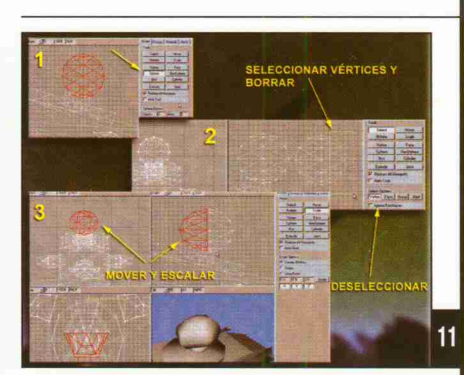
A partir de cilindros con tres divisiones, obtenemos los cañones.



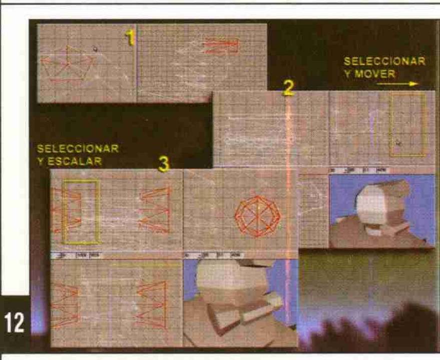
Dos piezas importantes, pero muy fáciles de realizar.



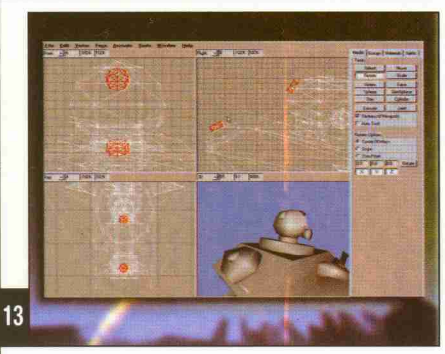
Una pieza de detalle para equilibrar la forma del casco de la nave.



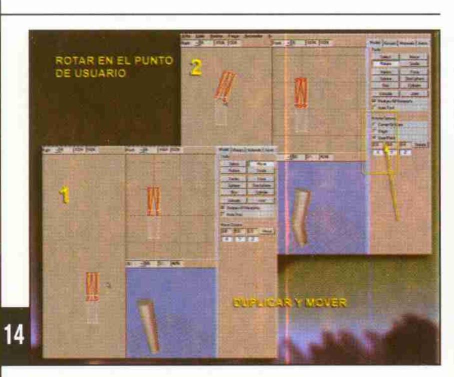
Hay que encajar bien el conector al casco mediante un buen escalado.



La cámara y el depurador de aire son dos piezas importantes para el aspecto humano al casco.



Los ajustes del tubo son piezas de detalle.



Primera forma de modelar el tubo.



La rotación de las piezas activando la opción "User Point" es bastante útil.

De la lista señalamos, por ejemplo, el primer elemento y en "Group" escribimos su nombre en el hueco de texto situado a la derecha del botón "Rename". Una vez realizada esta acción, pulsamos sobre"Rename" para completar el proceso. Cuando tengamos todas las piezas renombradas, el siguiente paso será asignarlas a cada uno de los botones numéricos que aparecen en "Smoothing Groups". Dejamos pulsado el botón "Assign", pero antes de asignar una pieza a un botón, hay que seleccionarlas. La elegimos de la lista y pulsamos sobre el botón "Select" en "Group", luego basta con pulsar el botón numérico que queremos asignar a esa pieza, por ejemplo, el "1". Antes de asignar otra nueva pieza es necesario deseleccionar la anterior pulsando de nuevo en "Select". Para los siguientes elementos realizaremos las mismas operaciones. Si queremos borrar todas las asignaciones pulsamos en "Clear All". En las opciones del apartado "Group", además de seleccionar, podemos ocultar ("Hide") y borrar ("Delete") cualquier pieza del modelo.

GUARDANDO EL MODELO

Antes de salvar el modelo en disco es importante tener claro el uso posterior que se le va a dar. Debemos saber en qué formato trabajaremos y si el modelo estará animado o no. Milkshape3D permite exportar a una multitud más que aceptable de formatos diferentes, englobando todas las necesidades posibles. Desde los ficheros de juegos como: *Quake, Unreal, Half Life* o incluso *Los Sims,* hasta los formatos 3D de las aplicaciones y motores gráficos más populares. En nuestro caso, diseñamos las bio-naves con posibilidad de que dispongan de ciertos movimientos. Aunque Milkshape3D permite la realización de esqueletos por puntos de unión a los modelos y dotarlos de movimiento, utilizaremos la aplicación Character FX para tal fin por disponer de más y mejores opciones. Teniendo en cuenta nuestros propósitos, debemos saber que Character FX importa objetos en formato .3DS, .OBJ (aunque puede importar también modelos en formato ASCII procedente del Milkshape3D -.TXT-). Además, para realizar y exportar el modelo con la animación, es conveniente disponer del objeto en un sólo *mesh* (objeto 3D), sobre todo si queremos exportar a .MD2. Así que nos vemos obligados a reagrupar todas las partes de la bio-nave en una sola antes de salvarlo. Para realizar esta operación, sólo tenemos que seleccionar todas las partes del modelo con "Select All" en el menú "Edit" y pulsar el botón "Regroup" en la pestaña "Groups" del panel de edición. Una vez agrupadas todas las partes, salvaremos el modelo en formato .OBJ. Debemos tener en cuenta que si antes de animar el modelo lo vamos a texturizar, no es conveniente entonces agrupar todas las partes en un solo grupo.

Ya tenemos nuestra bio-nave modelada y lista en el disco duro para completar su desarrollo. Pero antes de animarla vamos a realizar el proceso de texturizado.



NOTA

Si no se va a implementar ningún tipo de animación a la bio-nave, puede ser guardada en formato .3DS.

En el próximo número...

... completaremos el aspecto de la bio-nave de combate pintándola con el programa Deep Paint 3D.

Cómo desarrollar los efectos especiales de sonido (III)

Hemos estudiado cómo grabar, editar y aplicar efectos a nuestros sonidos. En esta entrega, completaremos el aprendizaje de Goldwave realizando operaciones muy interesantes.

EL EVALUADOR DE EXPRESIONES

Goldwave nos permite crear y manipular sonidos por medio de funciones matemáticas, muy útil para desarrollar nuestros propios efectos. Realicemos una práctica para conocer el funcionamiento de esta potente herramienta. Creamos una plantilla nueva de 5 segundos y elegimos la opción "Expresión Evaluator" situada en el menú "Tools". Podemos observar que básicamente se trata de un sistema para escribir fórmulas y darles un nombre. Vamos a fabricar el sonido de un motor. Partiremos de la onda de sonido de la función matemática coseno que varía en altura (eje Y). Matemáticamente la fórmula sería *Y= Cos(PI*F*T)*, donde F es la frecuencia y T el tiempo.

Vamos a implementar esta ecuación pero con una pequeña variación, así que, en la zona "Expresión" escribimos *cos(f*t)*(2*pi)*. En "Variables" cambiamos el valor de la frecuencia "f=" a 80 y el tiempo "t=" a 10. Pulsamos en el botón "Start" y listo. Al reproducir el sonido oiremos un motor. Jugando con la frecuencia obtendremos sonidos más agudos o graves. Por ejemplo, probad a colocar un valor igual a 10000 (Fig. 1).

También podemos crear efectos con esta herramienta, personalizando aún más nuestros sonidos.

EDITAR DIRECTAMENTE LA FORMA DE ONDA

Goldwave permite al usuario actuar directamente sobre la forma de onda. Esta opción posibilita tener una edición más exhaustiva del sonido y realizar cambios al milímetro dibujando con el ratón. Antes de poder actuar sobre la onda del sonido es necesario hacer zoom hasta alcanzar un nivel 2:1 o 5:1 (Fig. 2).

Para realizar una práctica, carguemos un sonido cualquiera que tengamos en el disco o grabemos uno nuevo.

Elijamos una zona del sonido y hagamos zoom hasta alcanzar un acercamiento 5:1. Al pasar el cursor del ratón sobre el dibujo de la onda, éste cambia de forma (una raya horizontal entre dos flechas verticales), indicando que ya es posible modificarla (Fig. 3).

Seguidamente, dibujamos con el ratón mientras hacemos clic con el botón izquierdo.

En la práctica, este tipo de modificaciones a tan bajo nivel puede ser útil para retocar interferencias, ruidos, picos de nivel o simplemente proporcionar algún tipo de efecto personal al sonido. (Fig. 4).

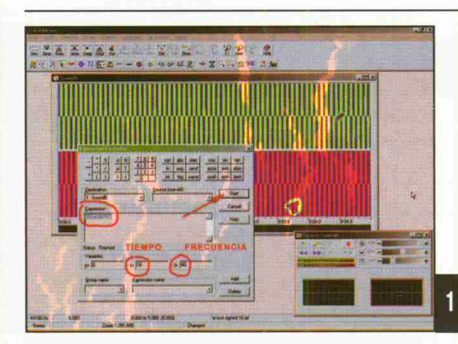
RESAMPLEAR

Este procedimiento nos proporcionará la opción de cambiar la

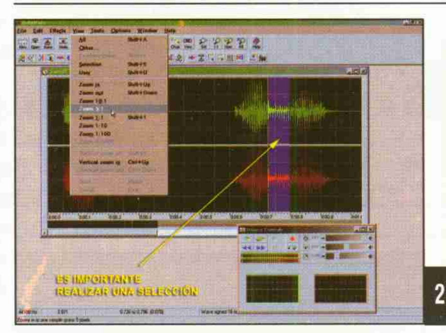


TRUCO

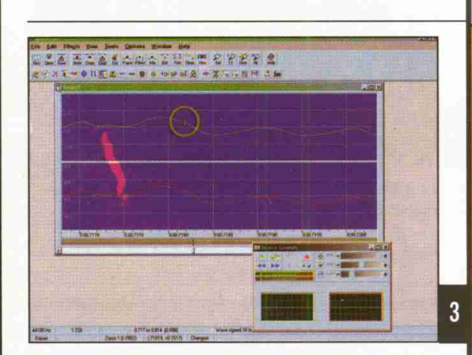
En la práctica, este tipo de modificaciones a tan bajo nivel puede ser útil para retocar interferencias, ruidos, picos de nivel o simplemente proporcionar algún tipo de efecto personal al sonido.



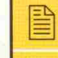
Procesos para la creación de un sonido desde cero con el evaluador de expresiones.



Para poder modificar manualmente la forma de onda es necesario hacer un zoom hasta un nivel 5:1.

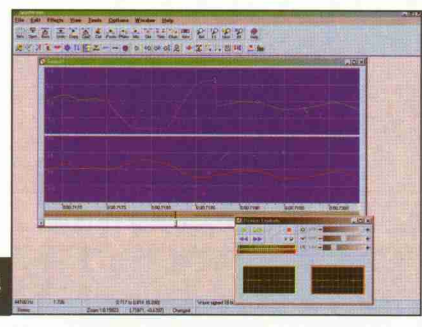


Para poder dibujar el sonido, el cursor debe cambiar de aspecto al pasar por encima de la forma de onda.

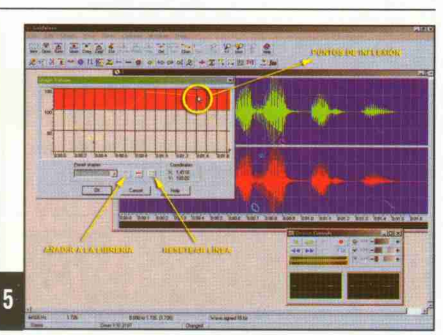


NOTA

Para desplazar la marca de comienzo de selección se pueden utilizar los cursores izquierdo y derecho + SHIFT y para la marca de final, cursores izquierdo y derecho + CTRL + SHIFT.

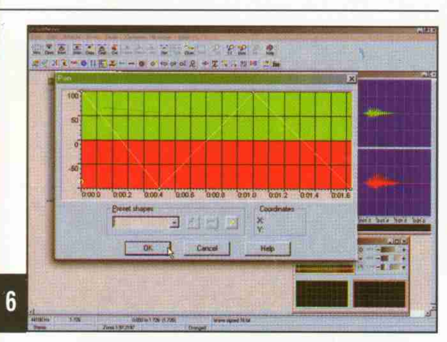

4

Podemos modificar la forma de la onda del sonido por medio del ratón.

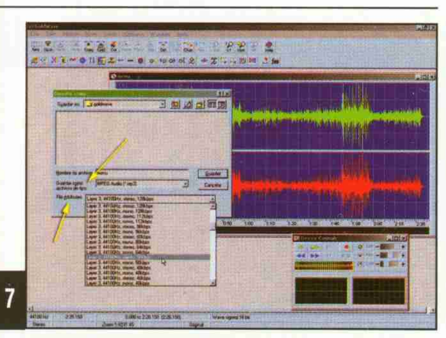


5

Para realizar un volumen personalizado en el sonido, podemos crear y editar cuantos puntos de inflexión deseemos.


6

La gráfica de la panorámica del sonido puede ser modificada con sólo utilizar el ratón.


7

Goldwave puede convertir el formato .WAV a .MP3 y viceversa sin ningún problema.

frecuencia de muestreo de nuestro sonido. Siempre es preferible, en un proceso de grabación y edición, utilizar la frecuencia más alta posible de muestreo. Una vez realizadas todas las operaciones al sonido es cuando se elige su calidad final. Es entonces cuando podemos utilizar esta función. Por ejemplo, si trabajamos con una calidad CD (44 Khz) y queremos obtener un resultado final a 22 Khz elegimos la opción "Resample" en el menú "Effects". Posteriormente, seleccionamos 22050 y "Ok".

Goldwave posee una opción muy interesante que permite reproducir un sonido a distinta velocidad "rate". La podemos encontrar en el menú "Effects" en "Playback Rate". Tiene el mismo funcionamiento que "Resample", pero sólo afecta a la duración y no a la calidad.

AJUSTE PERSONALIZADO DEL VOLUMEN

Hay un procedimiento muy útil para realizar un ajuste más personalizado del volumen del sonido, y es cambiando su envolvente de amplitud a lo largo del tiempo de duración.

En Goldwave encontramos esta función en el menú "Effects" en la opción "Shape" del submenú "Volume". Aparece una gráfica con la medida de volumen en el axis vertical y la duración en el horizontal. Por defecto, aparece el volumen normal en 100, representado por una línea horizontal amarilla. La parte de color roja representa un aumento del volumen y la parte sin color una reducción. Para crear puntos de inflexión en la línea de volumen, es suficiente con hacer clic con el ratón en cualquier lugar de la gráfica. Para mover cualquier punto, nos situamos encima de él y movemos el ratón manteniendo el botón izquierdo pulsado. Si por el contrario deseamos borrar un punto de inflexión, nos situamos sobre él y pulsamos el botón derecho del ratón (Fig. 5).

Como se puede apreciar, esta función es muy útil para dotar a las muestras de cierta expresión y lograr efectos más detallados.



TRUCO

Para cambiar los colores de la ventana de edición tenemos la opción "Colours" en el menú "Tools".

DEFINICIÓN

▶ ENVOLVENTE DE AMPLITUD
La envolvente de amplitud de un sonido es la representación gráfica de la dinámica de su volumen en el tiempo.

AJUSTE PERSONALIZADO DEL PANORÁMICO

De igual forma, podemos actuar directamente en el panorámico del sonido dibujando su evolución en el tiempo a través de una gráfica. Encontramos esta estupenda opción en el menú "Effects" en "Pan", situado en la opción "Stereo". La forma de actuar sobre la gráfica es exactamente igual que en la anterior opción. La única diferencia radica en la gráfica en sí. En ella se representa, con dos colores, cada uno de los canales estéreos del sonido. El color rojo indica el canal derecho y el verde el izquierdo. Podemos entonces dibujar el paso del sonido de un altavoz a otro a nuestro gusto. Probemos, por ejemplo, dibujando la gráfica como se muestra en la figura 6.

Observamos cómo al reproducirlo, va pasando de un altavoz a otro. Jugando con esta opción, podemos obtener sonidos más envolventes y con más cuerpo. El límite está, como siempre, en la imaginación.

En el próximo número...

... realizaremos los sonidos de "Zone of Fighters", empezando por crear los disparos, explosiones y otros efectos.

Gráficos 2D y sprites

Después de imprimir texto y mapa de bits, vamos a continuar aprendiendo más sobre los gráficos en el mundo 2D de Blitz3D.

Estudiaremos otros aspectos del búfer de pantalla, así como el manejo de primitivas gráficas en 2D y cómo trabajar con los píxeles (puntos de color) de la pantalla. Terminaremos explicando el funcionamiento de los sprites dentro de un entorno 3D.

ALGO MÁS SOBRE LOS BÚFERES

En el número 5 explicamos el funcionamiento de los búferes de vídeo y de la técnica del doble búfer. Continuando con este tema, encontramos instrucciones en Blitz3D que nos permiten cargar una imagen directamente a cualquiera de los dos búferes o salvarlos a disco. La función de cargar es *LoadBuffer (búfer, fichero de imagen $)* y la de salvar ya la conocemos: *SaveBuffer (búfer, fichero de imagen $)*.

En Blitz3D también podemos realizar operaciones de gran velocidad con píxeles o puntos de color de la pantalla. Para poder trabajar a este nivel es necesario bloquear el búfer en donde se realizan estas operaciones mediante la instrucción *LockBuffer nombre del búfer*. Para desbloquearlo utilizaremos *UnlockBuffer nombre del búfer*. Una vez bloqueado el búfer, sólo es posible realizar operaciones gráficas con píxeles.

OPERACIONES CON PÍXELS

Para leer un valor de color, es decir, un número que contiene el canal alfa, el color rojo, verde y azul en un punto de la pantalla, se utiliza la función *ReadPixel (X,Y,búfer)*.

Esta instrucción es muy similar a *GetColor X,Y*. La diferencia es que *GetColor* extrae las componentes del color (rojo, verde y azul) y la pasa a las funciones *ColorRed() (componente rojo), ColorGreen() (componente verde)* y *ColorBlue() (componente azul)*.

(Ver funcionamiento en el ejemplo 1 del numero 6: ejemplo6_1.bb).

Existe una función que nos permitirá leer un píxel de la pantalla a mucha mayor velocidad, se trata de *ReadPixelFast (X,Y,búfer)*. Funciona exactamente igual que su hermana *ReadPixel()*, sin embargo es conveniente no olvidarse de bloquear el búfer antes de utilizarla y desbloquearlo después de su uso. Si no lo hacemos así, la función no funcionará y además podría provocar el cuelgue del sistema.

De igual forma podemos escribir un píxel en pantalla con las funciones *WritePixel (X,Y, búfer)* y *WriteFastPixel(X,Y,búfer)* (ver "ejemplo7_1.bb").

También podemos copiar un píxel de un lugar de la pantalla a otro con las instrucciones:

```
CopyPixel (X origen, Y origen,
búfer origen, X destino,
Y destino, búfer destino)
```

y su equivalente, con más rapidez de resultados:

```
CopyFastPixel (X origen, Y origen,
búfer origen, X destino,
Y destino, búfer destino)
```

Antes de terminar, comentar que para cambiar el color de todas las operaciones de dibujo se utiliza la instrucción *Color Rojo, Verde, Azul*.



Primitivas 2D que puede generar Blitz3D.

PRIMITIVAS 2D

Es interesante también el uso de primitivas gráficas 2D en los videojuegos. Nos referimos a los puntos en la pantalla, líneas, cuadrados o círculos (ver "ejemplo7_2.bb") (Fig. 1).

Para dibujar un punto en la pantalla utilizaremos la instrucción *Plot X,Y*. Para dibujar una línea:

```
Line X comienzo de línea,
Y comienzo de línea, X final
de línea, Y final de línea
```

NOTA

Podemos también bloquear el búfer antes de usar *ReadPixel()* y *WritePixel()* para realizar las operaciones instantáneamente. Sin olvidar que siempre hay que volver a desbloquearlo una vez acabadas estas operaciones.



NOTA

Para saber la memoria de video total de la tarjeta gráfica usaremos la función *TotalVidMem()* y para saber la que queda *AvailVidMem()*.

Si lo que deseamos es dibujar un rectángulo usaremos:

```
Rect X,Y, Longitud, Altura,
1 (relleno) / 0 (transparente)
```

Y para dibujar un círculo o elipse utilizaremos:

```
Oval X,Y, Longitud, Altura,
1 (relleno) / 0 (transparente)
```

OTRAS FUNCIONES GRÁFICAS

Vamos a estudiar algunas funciones muy interesantes que Blitz3D posee para el tratamiento gráfico de la imagen, que nos ayudarán a crear efectos de scroll o sistemas de visualización 2D muy utilizado en la mayoría de los videojuegos.

CREANDO VENTANAS DE VISUALIZACIÓN

Con esta función podemos definir el área de la pantalla que se visualizará; es decir, crea una ventana sobre el fondo y sólo se verá el fondo a través de ella, el resto es ignorado.

Este tipo de procedimiento visual es muy utilizado en juegos donde la acción se desarrolla en una porción de la pantalla, dejando libre el resto para otros tipos de gráficos, como puede ocurrir en algunos juegos de Rol (Ver Fig. 2).

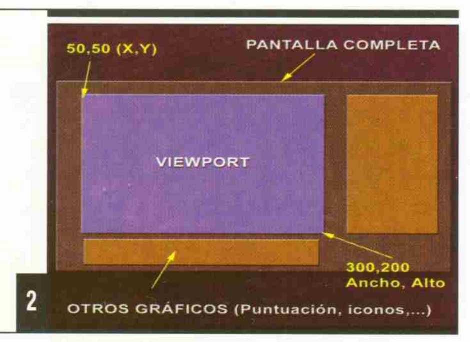
Aunque el concepto parece simple, la utilización de esta función es algo confusa. Así que vamos a ver detenidamente cómo funciona y cómo debemos utilizarla.

```
ViewPort Coordenada X,
Coordenada Y, Tamaño horizontal,
Tamaño vertical
```

Las coordenadas X e Y indican dónde está situada la esquina superior izquierda de la ventana y el tamaño horizontal y vertical hasta donde se extenderá. Es importantísimo tener en cuenta que es necesario borrar una ventana creada en el caso de definir otra nueva. Para realizar esta operación basta con definirla como si fuera la pantalla completa; es decir, si definimos un modo gráfico de 800 x 600, debemos escribir "ViewPort 0,0,800,600" para borrar la vista.

En el ejemplo "ejemplo7_3.bb" (Fig. 3) se puede observar un efecto realizado con esta función, en donde se crea e inicializa continuamente una ventana de visualización, generando así una ventana movible sobre un fondo fijo.

La siguiente cuestión que nos podemos plantear es cómo conseguimos tener independientemente una vista definida en donde se muestra un fondo (zona de juego) y unos gráficos que representen, por ejemplo, la puntuación u otra actividad (un buen ejemplo de esta situación la podemos encontrar en cualquier juego de acción donde se muestre en una esquina de la pantalla un radar). Pues bien, para resolver este problema debemos establecer un orden específico a la hora de visualizar los gráficos. En la figura 4 se muestra el esquema de este orden.



Un esquema de ejemplo de interfaz de juego utilizando ventanas.



Utilización de ventanas móviles para mostrar porciones del fondo.

Bucle

- Inicializar Ventana
- Borrar pantalla
- Dibujar otros gráficos
- Definir Ventana
- Dibujar Fondo o accion del juego
- Intercambiar búfers

Fin Bucle

4

Esquema del orden a seguir para visualizar fondos y otros gráficos utilizando ventanas.

En el ejemplo "ejemplo7_4.bb" se puede observar cómo funciona este procedimiento.

CREANDO UN SCROLL PARALLAX

La implementación de sistemas de scroll de pantalla en un juego es una tarea que puede tomar muy diversos caminos. La manera más sencilla, aunque no la más rápida y eficaz, es sin duda el movimiento completo de los fondos. Generalmente esta técnica es utilizada para scrolls cíclicos, es decir, el fondo se repite continuamente, por ejemplo, cuando sale por la izquierda va apareciendo por la derecha de la pantalla. Blitz3D posee algunas funciones específicas para este tipo de procedimientos: "TileBlock" y "TileImage":

```
TileBlock Imagen [,Coordenada X,
Coordenada Y, fotograma de la
imagen a usar]
TileImage Imagen [,Coordenada X,
 Coordenada Y, fotograma de la
imagen a usar]
```

Estas funciones realizan un tileado (repetición) de la imagen utilizada para los campos de scroll, además de poderlas mover. La única diferencia entre una función y otra es que "TileBlock" ignora

el color utilizado para la transparencia y "TileImage" no. En el ejemplo "ejemplo7_5.bb" se muestra un ejemplo muy sencillo de scroll múltiple (parallax) utilizando estas dos funciones.

Es importante tener en cuenta el orden de utilización de "TileBlock" y "TileImage" para lograr el efecto deseado.

SPRITES

Sabemos que Blitz3D trata a los elementos que forman el mundo 3D (luces, terrenos, modelos) como entidades. Pues bien, los sprites en Blitz3D son también entidades planas de forma cuadrada o rectangular con sólo dos polígonos. Por defecto, este objeto plano siempre mira hacia la cámara, dando la sensación de que es un objeto en 3D y además es tratado como si lo fuera.

CREANDO SPRITES

El uso de sprites es muy útil para simular explosiones, humo o disparos en un entorno 3D. Al ser tratado como un objeto 3D o entidad es posicionado en la pantalla a través de las coordenadas X, Y y Z. En el ejemplo 3 ("ejemplo7_6.bb") del CD creamos un sprite y lo colocamos delante de la cámara. Además, con la tecla "1" activamos o desactivamos el modo *wireframe* (alámbrico, sin textura). De esta forma podremos observar la forma y los polígonos que forman un sprite. Para crear un sprite nuevo, sólo es necesario una variable que lo contendrá (Fig. 5):

```
Sprite = CreateSprite ()
```

Como comentamos, esta entidad llamada Sprite es tratada como un objeto 3D, así que podemos aplicarle los mismos efectos y posicionarla en el entorno 3D. Luego, si queremos, le podemos aplicar una textura para darle imagen. Pero hay una forma de realizar estas dos operaciones a la misma vez, y es con la instrucción:

```
LoadSprite (imagen_textura$
[, modo de aplicar textura]
[, entidad madre]
```

Este comando te permite cargar una imagen para crear un sprite con ella. En realidad, lo que hace es crear un sprite y asignarle como textura la imagen cargada. Además, opcionalmente es posible asignar la textura de diferentes modos (opción "[,modo de aplicar textura]"). Así disponemos de las siguientes opciones (ver "ejemplo7_7.bb") (Fig. 6):

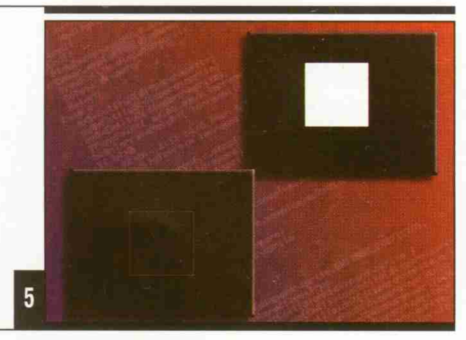
- **Valor 1: Color del mapa.** Es el valor por defecto y se muestra como es.
- **Valor 2: Canal Alfa.** Si la textura tiene canal alfa, las zonas alfas de ésta se harán transparentes.
- **Valor 4: Máscara de color.** Todas las áreas de la textura con color negro (0,0,0) no se dibujarán.
- **Valor 8: Mipmapped.** Se muestra una versión de la textura con baja calidad a grandes distancias.
- **Valor 16: Ajuste U** (horizontal) de la textura para evitar enrollamientos.
- **Valor 32: Ajuste V** (vertical) de la textura para evitar enrollamientos.
- **Valor 64:** Se utiliza para realizar un mapeado del entorno con brillo.

Generalmente, las más comunes suelen ser los valores 1 y 2, sobre todo si utilizamos sprites para representar fuego, humo, etc.

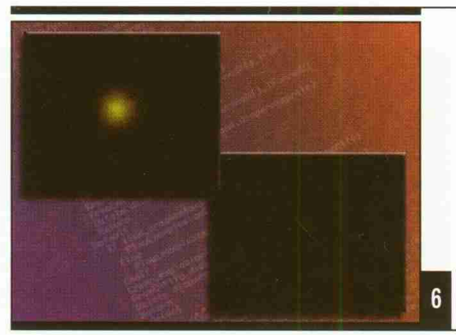
La siguiente opción ([,entidad madre]), permite asignar al sprite una entidad madre, esta cualidad es muy útil, ya que si movemos o escalamos la entidad madre el sprite lo hará también. Sin embargo, si movemos el hijo (sprite), no afectará a la entidad madre. Por ejemplo, el protagonista del juego (entidad madre) lleva una piedra (entidad hijo, sprite) en la mano. Cuando el protagonista se mueve la piedra se moverá con él y seguirá en la mano. Al arrojar la piedra, se desplaza el sprite "piedra", pero el protagonista no se mueve con ella, se queda en la misma posición.

RECORDATORIO

Un sprite es una imagen (generalmente de pequeño tamaño) usada para gráficos animados o simplemente para un icono.



En la imagen se puede comprobar que un sprite está formado por dos polígonos planos.



Un ejemplo de textura para sprite con la opción de máscara activada.



En la salida del ejemplo 5.bb, se puede comprobar la dependencia de una entidad hijo hacia su entidad madre.

En el ejemplo 5 ("ejemplo7_8.bb") del CD se puede observar cómo al mover el cubo (entidad madre) arrastra al sprite (entidad hijo) consigo, mientras que si se mueve el sprite, el cubo se mantiene inmóvil (Fig. 7):

```
...
cubo=CreateCube()
sprite=LoadSprite("c:\particle
.bmp",1,cubo) ←Sprite, modo
texturizado, padre/madre
PositionEntity Sprite,0,2,0←
PositionEntity cubo,0,0,15←
...
```

Al crear el sprite, le asignamos como madre el cubo. Posicionamos el sprite arriba del cubo para que se pueda ver. Observar que sólo tenemos que desplazar la coordenada Y del sprite y no la Z, ya que al ser hijo del cubo, posicionando éste se posicionará el sprite con él.

MANIPULANDO SPRITES

Aunque pueden ser controlados como una entidad, los sprites tienen algunos comandos propios para rotarlos o cambiar su tamaño (ver "ejemplo7_9.bb") (Fig. 8):

Con *RotateSprite sprite, angulo de rotación #* podemos rotar un sprite desde su eje y con *ScaleSprite sprite, escala horizontal#, escala vertical#* podemos cambiar su tamaño.

Para cambiar el eje de rotación del sprite utilizaremos la instrucción:

```
HandleSprite sprite,
desplazamiento horizontal#,
desplazamiento vertical#
```

Esta instrucción también nos sirve para desplazar el sprite (ver "ejemplo7_10.bb") (Fig. 9).

MODOS DE VISTA

El problema que surge cuando se visualiza un objeto plano, como un sprite, en un entorno 3D, es cómo controlar su orientación con respecto a la cámara. Por ejemplo, supongamos que tenemos un juego; en él se muestra un terreno en 3D con muchos árboles. Estos árboles, para que el juego tenga un buen rendimiento, son realmente sprites, es decir, un objeto plano con el dibujo de un árbol (textura). Si orientamos los árboles siempre hacia la cámara, podremos ocultar que en realidad se trata de un dibujo plano y no de un objeto en 3D. Esta técnica fue la utilizada en los antiguos juegos en 3D, en donde los enemigos y objetos de decoración eran sprites orientados hacia la cámara.

Blitz3D permite controlar de cuatro formas diferentes de orientación con respecto a la cámara. Para activar esta operación se utiliza la instrucción:

```
SpriteViewMode sprite,
modo de orientación.
```

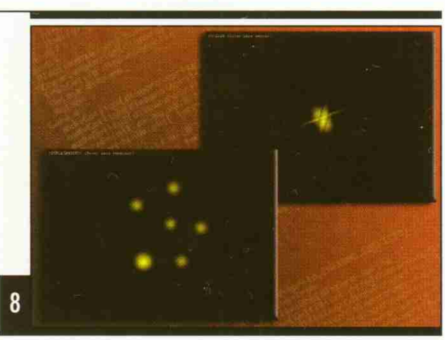
Por defecto se encuentra activado el modo 1, en el cual el sprite siempre está mirando a la cámara. El modo 2 se mantiene al sprite libre de orientación. Con el modo 3 se consigue una orientación hacia la cámara, pero no cuando ésta se inclina, y en el modo 4 se consigue una orientación completa con todos los movimientos y giros de la cámara.

En el ejemplo 8 ("ejemplo7_11.bb"), se puede observar el funcionamiento de esta importantísima función (Fig. 10).

A los sprites se les puede aplicar todos los controles, como a cualquier otra entidad. Así, es posible detectar colisiones con ellos o entre ellos, moverlos por el entorno 3D, manipular sus características, aplicar efectos y obtener información de su estado.

RECORDATORIO

Blitz3D define a todos los elementos 3D como entidades. Este sistema facilita un método de asignaciones muy útil que permite controlar el estado, el movimiento o las colisiones de luces, cámaras o superficies como si fueran objetos.

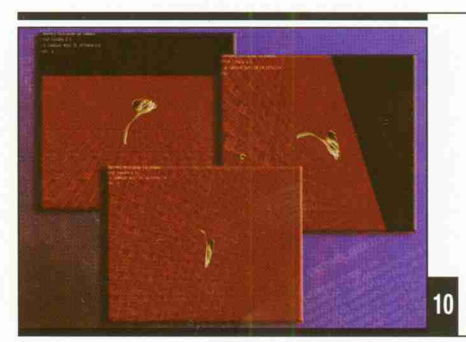

8

Algunos efectos interesantes con las opciones de escalado o desplazamiento de un solo sprite.



9

La función HandleSprite puede desviar el eje de rotación de un sprite.


10

Distintos comportamientos en el modo de vista de un sprite frente a la cámara.

En el próximo número...

... explicaremos todas estas operaciones, comunes entre todas las entidades. Aprenderemos todo lo necesario sobre este sistema utilizado por Blitz3D y así prepararnos más a fondo para manipular el mundo 3D y todas sus funciones.

Editores de audio. Sound forge 6.0 (I)

En este número empezamos una serie de tutoriales básicos dedicados a los editores de audio más populares del mercado.

Para comenzar, hemos querido tener aquí al que quizás sea uno de los más potentes, Sound Forge 6.0. Por lo general, la forma de usar los editores de audio es muy similar, así que nos centraremos sólo en las nuevas posibilidades que nos podamos encontrar en cada uno de ellos.



SOUND FORGE 6.0

Sound Forge es de la desarrolladora Sonic Foundry, especializada en aplicaciones para el control de audio. Este programa siempre se ha caracterizado por tener una interfaz de usuario muy cómoda, flexible e intuitiva desde la primera versión. Es ampliamente utilizado en el mundo amateur y profesional por su versatibilidad y potencia. Una característica importante es la posibilidad de poder tener varios sonidos cargados a la vez en ventanas diferentes y trabajar con calidades de hasta 32 bits a 192 Khz. Pero quizás lo que ha distinguido este programa de los demás es su velocidad de trabajo en procesos de edición como cortar, pegar o mover y en la posibilidad de obtener previos en tiempo real de los procesos con efectos antes de aplicarlos. Además, permite visualizar ficheros de vídeo para poder así sincronizar, por ejemplo, el audio para una película. Puede trabajar con multitud de formatos de audio como .WAV, .MP3, .OGG (OggVorbis), .AIF, .SND, etc.



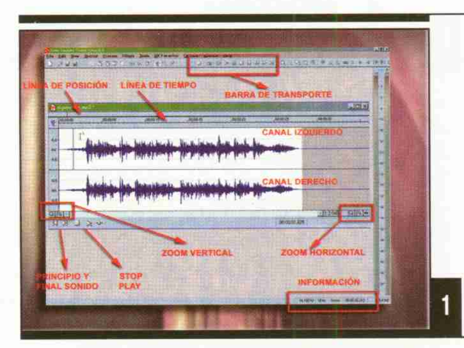
TRUCO

Para mover la línea de tiempo podemos utilizar los cursores derecha e izquierda o CTRL + dichos cursores. Además , para realizar una selección podemos mover la línea de tiempo con el método anterior pero manteniendo pulsada la tecla SHIFT.

CARGAR, REPRODUCIR Y EDITAR UN SONIDO

Podemos cargar un sonido de dos formas diferentes. Una manera sería sólo para reproducirlo, sin posibilidad alguna de editarlo. Para ello sólo tenemos que activar la casilla "Open as read-only" en la ventana "Abrir". Si activamos esta casilla, cargaremos el sonido y podremos entonces editarlo. Una vez cargado, se abrirá una ventana con la gráfica del sonido. En la figura 1 se describe cada uno de los elementos que componen la ventana de edición.

Para reproducir el sonido, podemos pulsar en el icono ▶, la barra espaciadora (también para parar) o bien desplazar el puntero del ratón haciendo clic sobre la barra situada sobre la línea de tiempo (Ver Fig. 1). Para seleccionar una porción del sonido es suficiente con utilizar el ratón, manteniendo pulsado el botón izquierdo mientras desplazamos el puntero. El principio y final de selección se pueden mover, independientemente, también con el ratón. Para seleccionar sólo uno de los canales es suficiente con



Descripción de los elementos del la ventana de edición.

hacer clic en la zona más externa del canal.

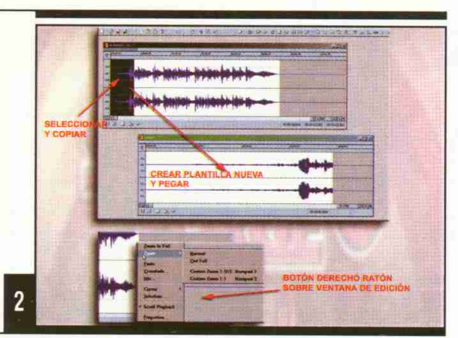
Cuando pulsamos el botón derecho del ratón encima de la ventana de edición aparecerá un menú flotante con diferentes opciones dependiendo de si hemos seleccionado o no una parte del sonido. En caso de no tener ninguna selección, podemos hacerlo en la opción "Selection" de dicho menú.

Para cortar un trozo del sonido basta con seleccionarlo y pulsar CTRL + X. Para insertar o copiar una porción de sonido de un lugar a otro seleccionamos lo que queremos copiar, pulsamos CTRL + C y llevamos la barra de posición, haciendo clic con el ratón, al lugar donde queramos colocarla. Posteriormente pulsamos CTRL + V. Es posible hacer esta misma opera-



NOTA

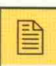
Si pulsamos el botón derecho del ratón fuera de la ventana de edición aparecerá un menú flotante con acceso rápido a las funciones de archivo y configuración del programa.



Procedimiento para crear un nuevo sonido a partir de una selección.

ción en otro sonido diferente u otra ventana de audio nueva. Seleccionamos un trozo de sonido y copiamos con CTRL + C. Pulsamos en "File \ New". Se abrirá una nueva ventana en blanco. Hacemos clic en cualquier lugar de esta nueva ventana y pulsamos CTRL + V. La porción seleccionada del primer sonido la hemos copiado a la nueva plantilla (Fig. 2).

Para realizar las operaciones de zoom disponemos de multitud de opciones. Bien mediante los iconos lupas situados a ambos extremos de la ventana de edición, a través del menú flotante de edición o por medio de las opciones de zoom del menú "View" (Fig. 2).

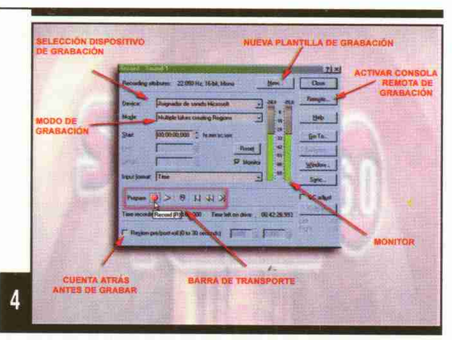


TRUCO

Podemos alejarnos o acercarnos a la gráfica del sonido de una forma más rápida por medio de los cursores. Cursor arriba para acercarnos y cursor abajo para alejarnos.



Diferentes maneras de cambiar el formato de un sonido.



Descripción de los elementos de la ventana de grabación.



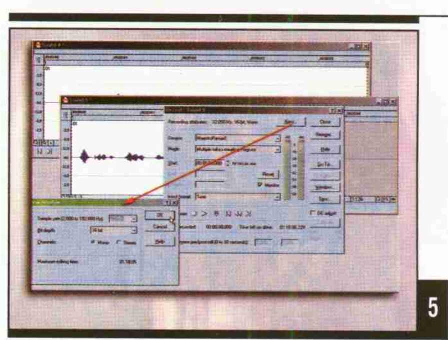
GRABAR UN SONIDO

Vamos a explicar de una forma breve el procedimiento para grabar un sonido desde un micrófono externo. Supongamos que queremos grabar una voz a 22 Khz, mono y calidad de 16 bits. Creamos una nueva plantilla en "File \ New". A continuación, debemos cambiar el formato de dicha plantilla. Hay varias formas, una primera, a través del menú flotante de edición en la opción "Properties" y otra, pulsando el botón derecho del ratón sobre la información correspondiente del fichero y eligiendo la opción deseada (Ver Fig. 3).

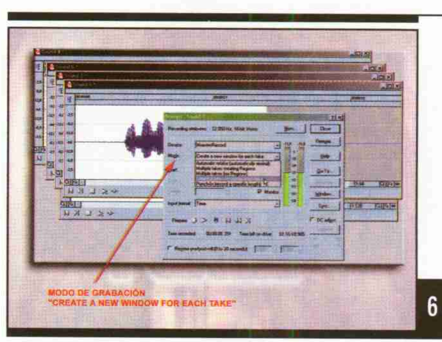
Una vez elegido el formato, pulsamos en el icono y aparecerá una ventana de opciones cuya descripción se muestra en la figura 4.

Dentro de ésta, el botón "New", nos permite abrir una nueva plantilla para grabar otro sonido sin tener que salir de la ventana "Record" (Ver Fig. 5).

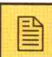
Además nos permitirá crear el nuevo sonido en otro formato. Sin embargo, la mejor opción, si queremos hacer grabaciones continuas con el mismo formato, es elegir "Create a new window for each take" en el modo ("Mode") de grabación. Cada vez que pulsemos el botón de grabar se creará una nueva plantilla automáticamente para alojar la nueva grabación. Este modo de trabajar ahorra mucho tiempo y esfuerzo y se consiguen grabaciones más uniformes (Ver Fig. 6).



Procedimiento para grabar nuevos sonidos con diferentes formatos desde la ventana de grabación.



Procedimiento para la creación y grabación continua de sonidos con el mismo formato.



TRUCO

Para grabar un sonido nuevo, hemos de asegurarnos de hacerlo en una habitación silenciosa, para que no se oiga ningún ruido de fondo. Es conveniente también no acercarse demasiado al micrófono.



En el próximo número...

... aprenderemos a crear sonidos nuevos, realizar "loops", listas de selecciones, y haremos una práctica creando el sonido de una explosión desde cero.

Evolución técnica de los **arcades**

La estructura de los arcades se basaba en más o menos grandes mapeados que conformaban todos los niveles del juego.

La evolución técnica de estos juegos pasa por la manera de representar este mapeado en pantalla, además, claro está, de la mejora de los gráficos, animaciones y sonidos. Por último, cabe destacar la revolución técnica que supuso el paso de la segunda a la tercera dimensión para este tipo de juegos.

Nos encontramos, de este modo, diferentes estilos de juegos de plataformas según la forma de representar el mapeado: pantalla fija, scroll de pantalla simple y scroll de pantalla múltiple.

JUEGOS DE PLATAFORMAS DE PANTALLA ESTÁTICA

En los comienzos del desarrollo de juegos de plataformas para PC, era complicado el uso del scroll de pantalla por limitaciones técnicas. En su lugar, se optó por el dibujo de las pantallas una a una, las cuales iban apareciendo a medida que el protagonista llegaba a los bordes de la misma o bien cruzando una puerta o un teletransporte. Estos juegos para PC adoptaron la forma de la mayoría de las producciones para los microordenadores más populares como el ZX Spectrum, MSX, Atari o Amiga. A veces, los mapeados de estos juegos eran grandes dibujos divididos en cuadrículas, las cuales hacían referencia a una pantalla.

Era muy común el uso de *tiles* para representar cada pantalla, ya que no se disponía de mucha cantidad de memoria para almacenar muchos gráficos diferentes. Las pantallas se formaban a partir de pequeños gráficos colocados estratégicamente, como si de un mosaico se tratara, para formar las paredes, suelos, etc.

Prácticamente todos los títulos desde 1986 hasta 1990 utilizaban esta técnica. Destacamos las producciones españolas *Livingstone Supongo* (Topo Soft, 1989) o *Goody* (Opera Soft, 1987), así como la serie de *Rick Dangerous* (Core, 1989-90).

EL SCROLL DE PANTALLA

El uso del scroll de pantalla se hizo cada vez más popular, al dar un aspecto visual más atractivo al juego y generar un tipo de jugabilidad más dinámica. Con este sistema, los mapas iban cruzando la pantalla en la dirección en que se movía el protagonista. Era un recurso muy común entonces el uso de los mapeados por *tiles*. Esta técnica ofrecía la posibilidad de generar grandes escenarios con pocos gráficos, ya que los *tiles* estaban diseñados tal que unidos formaban un patrón ampliable sin fin.

Fue el juego *Super Mario Bros* el que prácticamente introdujo esta técnica para la consola de Nintendo. Por entonces, el PC no podía visualizar más de 16 colores a la vez y no era demasiado potente para representar decentemente un scroll por *tiles* en un monitor. Sin



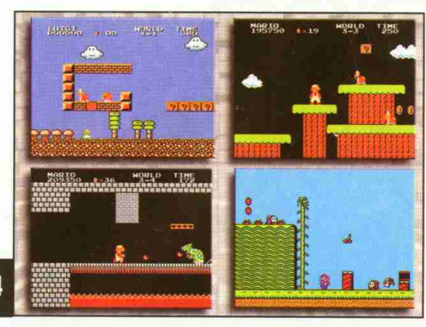
1 *Dos ejemplos del mapeado de un juego de plataformas.*



2 *El mapeado de "Cauldron 2" disponía de cientos de pantallas que unidas formaban un castillo.*



3 *Goody y la serie Rick Dangerous eran dos claros ejemplos de plataformas de pantalla estática.*



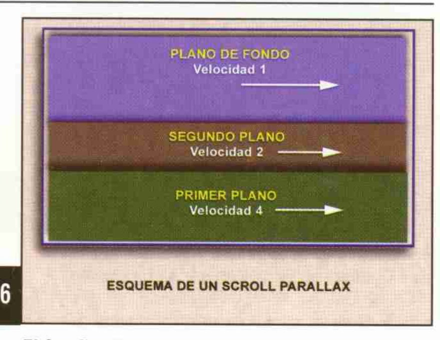
4

El Super Mario Bros de la consola Nintendo introdujo el concepto de plataformas por tiles que luego fue adoptado por multitud de juegos de PC.



5

Risky Woods fue un magnífico ejemplo de plataformas con doble scroll.



6

El Scroll múltiple trajo profundidad y dinamismo a los escenarios de los juegos de plataformas.



7

Pandemonium constituyó el primer paso de los juegos de plataformas en 2D al mundo 3D.

embargo, sale al mercado una serie de juegos llamados *Commander Keen* (1990) de una empresa nueva llamada ID Software, que utilizaba scroll por tiles. Muchos fueron lo que le siguieron, sobre todo arropados por nuevas tecnologías que poco a poco alimentaban al popular compatible del IBM PC.

SCROLL PARALLAX

Ya en 1991 se empezó a trabajar masivamente con técnicas de scroll más complejas. Por ejemplo, se utilizó un doble scroll para simular el movimiento del cielo como en *Risky Woods* (Dinamic, 1992) o *Prehistorik 2* (Titus, 1992).

Y poco a poco se pudieron mezclar varios scrolls a la vez. Nace así el scroll *parallax* o de paralelaje, consistente en mezclar varios planos de decorado desplazándose a distinta velocidad para simular profundidad.

Se mezclaban fondos móviles que simulaban el cielo o montes y mapeados por *tiles* en los primeros planos de scroll. Se lograba así un realismo extremo similar a las recreativas, y pronto todos los títulos del género optaron por usarlo.

Aunque estas formas de representación iban unidas al avance técnico, significaban también un estilo de juego, y que títulos que no entendían de sistemas o modas, mantuvieran uno u otro método a favor de la jugabilidad. Es el caso de juegos como *Another World* o *Flashback*, que si hubieran optado por el paso de pantalla mediante scroll habrían perdido todo su encanto estratégico.

EL PASO AL 3D

La evolución tecnológica, una vez más, dio la oportunidad a los desarrolladores de arcades de plataformas de investigar nuevas tendencias de visualización. Fue el caso del estancamiento que sufrió la eterna e inamovible segunda dimensión frente al mundo 3D. Este fenómeno ocurrió, sin embargo, en las consolas con el juego *Crash the Bandicoot* y *Mario 64* de manos de Nintendo. Para el PC se empezó a realizar títulos utilizando el 3D con resultados bastantes aceptables para la época. Un buen ejemplo y quizás el primero de este tipo lo tenemos en la conversión del juego *Pandemonium* (*Crystal Dynamics*, 1996) de PlayStation para el PC. Soportaba resoluciones de hasta 640 x 480 y gozaba de todo lujo de detalles gráficos. Realmente se trataba de un juego en 2.5 D (dos dimensiones y media) ya que el personaje se movía como si fuera en 2D, es decir, como en todos los juegos de plataformas (correr hacia delante y atrás y saltar), pero todo el entorno era en auténtico 3D donde la cámara seguía la acción automáticamente.

En otros títulos de calidad, como el clásico *Rayman*, se optó por seguir la serie adaptando el juego a los nuevos tiempos 3D. Surgió entonces el maravilloso titulo *Rayman 2: The Great Scape* (*Ubi Soft*, diciembre del 99).

Hoy día, se realizan pocos juegos de plataformas como antaño. Quizás *Abe´s Oddysee* y su continuación *OddWorld Abe´s Exoddus* (*Oddworld Inhabitants*, 1998) (posiblemente los mejores plataformas de la historia del PC), sean un claro ejemplo de que este género tiene mucha guerra que dar todavía.

En el próximo número...

... hablaremos de un tipo de arcade que está siempre presente en la vida de cualquier jugador: los matamarcianos o *shoot´em up*.

Cuestionario Videojuegos 7

Preguntas

1. ¿Cómo podemos cambiar el color de un píxel de la pantalla?

2. Describe la función que nos permite orientar un sprite hacia la cámara de diferentes maneras.

3. Escribe un grupo de sentencias que nos permita mover la cámara mediante el ratón.

4. ¿Qué funciones nos permitirán, por ejemplo, mover un objeto por un terreno utilizando el ratón?

5. ¿Cómo podemos unir dos piezas en MilkShape 3D?

6. ¿En qué formato podemos guardar el modelo para utilizarlo en CharacterFX?

7. ¿Para que nos puede servir utilizar la función de "resamplear" en GoldWave?

8. ¿Cómo podemos conseguir que el sonido pase de un altavoz a otro?

9. En el programa Sound Forge 6.0, ¿qué modo de grabación debemos usar para realizar grabaciones diferentes con el mismo formato sin salir de la ventana de grabación?

10. En Sound Forge 6.0, ¿cómo podemos crear un sonido desde cero sin realizar ningún tipo de grabación?

Respuestas al cuestionario 6

▷ **1.** La manera más rápida sería aplicando el comando "Replace":

```
frase$="Mi coche es verde"
Replace(frase$,"es ","")
```

Sustituimos "es " por nada.

▷ **2.**
```
ImageCollide (imagen_raton, posX_raton, posY_raton, 0,
imagen_boton, posX_boton, posY_boton, 0)
```

▷ **3.** Por orden irían: logotipo distribuidor, logotipo desarrollador, logotipo del juego, argumento del juego. Necesarias serían el logotipo del distribuidor y del desarrollador.

▷ **4.** Un menú se puede dividir en: opciones gráficas, opciones de audio, opciones de control y opciones de juego.

▷ **5.** Moviendo las líneas discontinuas que marcan el comienzo y final de la letra.

▷ **6.** Para las edificaciones se utilizará el formato .3DS y para la bionave de combate se utilizará .3DS y si le aplicamos animación, . MD2 o . B3D.

▷ **7.** Aplicando un proceso de normalizado, es decir, aumentando al máximo el volumen sin llegar a saturar.

▷ **8.** Por medio del evaluador de expresiones.

▷ **9.** Una forma es en el "Track info" de la ventana de arreglo y otra a través del editor midi GM/GS/XG.

▷ **10.** La técnica de la cuantización.

Contenido CD-ROM 7

▶ AUDIO

■ Dance eJay 4.0
Realiza tu propia música de baile y sus correspondientes efectos visuales.

■ Beat 2000 Online 1.07
Podrás crear tu propia música para PC, aunque no tengas ni idea de composición.

■ Midi Quartet 1.0
Compón, graba, edita y reproduce gracias a este compositor de 12 canales.

■ Ear Trainer 1.0
Entrena tu oído musical con esta divertidísima pero útil herramienta.

■ Jambient 0.9
Usando el ratón o un joystick podrás componer asombrosas melodías.

■ Soundtrack Producer 1.01
Ponle banda sonora a tus vídeos gracias a este programa.

■ Soundforge 6.0



La aplicación que tratamos en el tutorial de este número. Potente y sencilla en su uso.

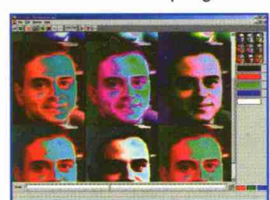
▶ DISEÑO 2D

■ NewView Graphics' File Viewer 6.0
Podrás ver imágenes en muy distintos formatos gracias a este programa.

■ Pixia 2.5
Edita imágenes fácil y cómodamente usando esta aplicación.

■ Adobe Type Manager 4.1
Demo del popular programa de Adobe de manejo de fuentes, en su versión en castellano.

■ TextPipe Pro 6.3
Añade complejos y vistosos efectos de texto sin necesidad de programar.



■ Ice Color 1.1
Potente herramienta para corregir colores y optimizar tus imágenes.

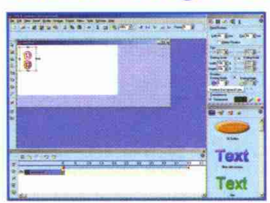
■ PhotoPlus 5
Crea y edita tus propios gráficos con la ayuda de esta completo programa.

▶ DISEÑO 3D

■ Xibit3D 1.0
Con esta aplicación podrás crear interesantes efectos 3D.

■ 3D Movie Maker
Crea fácilmente películas en 3D con este programa y da rienda suelta a tu creatividad.

■ 3D Gif Designer 2.2



Si en algún momento quieres hacer una web para mostrar tu juego, esta herramienta te será de gran ayuda.

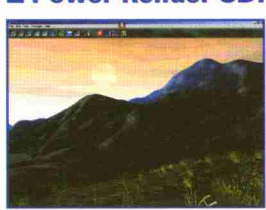
■ 3D Textures and Wallpapers
Una colección muy útil de texturas en tres dimensiones.

■ StyleSkin 2.0
Diseña y renderiza interfaces gráficas en dos o tres dimensiones.

▶ PROGRAMACIÓN

■ Mioplanet Pixel Ruler 133
Gracias a esta utilidad podrás saber la posición exacta de un objeto dentro de la pantalla.

■ Power Render SDK 4.0



Esta potente solución para desarrolladores de juegos tiene un montón de opciones y herramientas.

■ LocWise 2.0
Kit de desarrollo que te permite proveer soporte multilingüe gracias a packs de lenguaje.

■ UCanCode Form ++ Class Library 4.0
Completa librería, indicada sobre todo para el dasarrollo de aplicaciones con gráficos. Incluye Visual Studio 6.0

■ REBOL 2.5.0.3.1.
Aplicación para programar multi-plataforma.

■ EmEditor 3.28
Este editor soporta la sintaxis de un gran número de lenguajes, y además es compatible con Windows XP.

▶ JUEGOS

■ Abandon Loader 0.8b
Excelente utilidad para configurar y reproducir juegos *abandonware* de DOS.

■ Lemmings 3D



Versión 3D del popular juego en el que debes llevar a los pequeños lemmings a un lugar seguro.

■ SimCity 2000 Network Edition 1.0
Maravilloso juego apto para todas las edades.

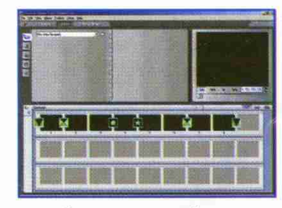
■ Walk Around 3d 1.0
Ejemplo hecho con Blitz3D.

■ Unreal Tournament 2003
Demo del popular juego 3D en su nueva versión que sin duda te entusiasmará.

■ Zone of fighters
De nuevo incluimos la nueva versión del juego, para que no te pierdas detalle.

▶ VÍDEO

■ Pinnacle Studio 7



Potentísimo programa muy sencillo de usar para editar vídeo y aplicarle impresionantes efectos avanzados.

■ Camtasia 3.0.2.
Graba, edita y reproduce vídeo de calidad de un modo muy profesional.

■ GlobFX Player 1.0.9
Herramienta muy completa con la que podrás ver y crear tus propios clips.

■ Windows Media Services 4.1.00
Descárgate los vídeos de mejor calidad gracias a esta interesante utilidad.

▶ EXTRAS

En este apartado encontrarás todos los ejemplos de los que hablamos en el coleccionable. Además, podrás encontrar dos *plugins* para Milkshape 3D y Character FX, para poder utilizar el formato .B3D.